한국군사과학기술학회지 제13권 제2호, pp. 171～179, 2010년 4월

학술논문 지상무기 부문

무인차량의 주행성능을 고려한 장애물 격자지도 기반의 지역경로계획

A Local Path Planning Algorithm considering the Mobility of UGV based on the Binary Map

이 영 일* 이 호 주* 고 정 호*
Young-il Lee Ho-Joo Lee Jung-Ho Ko

Abstract

A fundamental technology of UGV(Unmanned Ground Vehicle) to perform a given mission with success in various environment is a path planning method which generates a safe and optimal path to the goal. In this paper, we suggest a local path-planning method of UGV based on the binary map using world model data which is gathered from terrain perception sensors. In specially, we present three core algorithms such as shortest path computation algorithm, path optimization algorithm and path smoothing algorithm those are used in the each composition module of LPP component. A simulation is conducted with M&S(Modeling & Simulation) system in order to verify the performance of each core algorithm and the performance of LPP component with scenarios.

Keywords : Autonomous Navigation, LPP(Local Path Planning), Path Optimization, Path Smoothing, Binary Map, UGV (Unmanned Ground Vehicle)

1. 서 론

체코의 극작가 Karel Capek가 Robot의 어원인 “강제노동”을 의미하는 “Robota”에 대해 최초로 언급한 1920년 이후 로봇관련 기술은 괄목할만한 진보를 이룩하였다. 특히 근래에 들어서는 3D 산업현장, 화성탐사와 같은 우주 개척사업 그리고 감시 및 정찰의 군사 분야처럼 인간의 접근과 개입이 어렵거나 불가능한 거친 외부환경에서 운용되는 무인시스템에 대한 필요성이 증가하고 있다. 무인시스템이 다양한 환경에서 주어진 임무를 완수하기 위해서는 여러 분야의 기술들을 필요로 하지만 가장 근본적인 기술은 안전하고 빠른 최적의 경로를 통해 주어진 임무지역까지 자율적으로 주행해가는 자율주행(Autonomous Navigation) 기술이다. 무인차량의 자율주행 기술은 무인차량에 장착된 센서의 탐지영역을 기준으로 전역경로계획(GPP : Global Path-Planning)과 지역경로계획(LPP : Local Path-Planning)과 같이 계층적으로 구성된다[1]. GPP는 숙고적인 계층으로 미리 제공된 DEM(Digital Elevation Map)/DSM(Digital Surface Map) 및 FDB(Feature Data Base)를

† 2009년 12월 1일 접수~2010년 2월 4일 게재승인
* 국방과학연구소(ADD)
책임저자 이메일 : 이호주(hojoolee@yahoo.com)

기반으로 산이나 호수와 같은 대규모의 지형적 특성 그리고 임무위험도를 고려하여 주어진 목표점까지의 합리적인 경로설정을 오프라인상에서 수행한다. LPP는 보다 반응적인 계층으로 다양한 센서를 통해 가장 최근에 획득한 대략 수십 미터 이내의 월드모델링 데이터로부터 지형 경사도 및 거칠기와 같은 지형정보와 다양한 장애물 정보를 추출하고, 이를 활용하여 GPP에서 생성한 다음 경유점까지의 이동경로를 안전성(Safety)과 안정성(Stability) 관점에서 실시간으로 설정한다.

본 논문에서는 무인차량의 자율주행시스템을 위한 장애물 격자지도 기반의 지역경로계획 기법에 대해 언급한다. 무인차량에 장착된 쌍안 카메라 및 레이저 거리측정기와 같은 지형감지용 센서로부터 입력받은 특정 크기의 지형에 대한 고도값을 활용하여 무인차량의 주행성 여부를 판단하는 장애물 격자지도를 생성한 후, 이를 기반으로 GPP 경로점 지향적이면서 장애물을 회피하는 지역경로계획을 수행한다. LPP와 관련된 기존의 대표적인 방법으로는 Wall Following기법[2], VFF(Virtual Force Field)/VFH(Vector Field Histogram)[3~5]기법 그리고 휴리스틱 탐색기법[6,7]이 있다. Wall Following 및 VFF/VFH 기법은 특정한 조건에 놓이면 지역경로계획에 실패하게 되는 지역 최소점(Local Minima) 문제가 발생한다는 단점이 있다. 또한 트리 탐색기법을 활용하는 휴리스틱 탐색기법은 이러한 문제점에 강인성을 보이나 과도한 CPU 타임과 메모리를 요구한다는 문제점이 있어 실시간성을 요구하지 않는 GPP에 주로 활용되어 왔다. 하지만, 근래에 들어 컴퓨터 H/W의 급속한 발전에 힘입어 지형감지용 센서의 탐지영역을 구성하는 수십 미터의 격자지도에서는 LPP의 실시간성을 확보하게 되었다.

본 논문에서는 가장 널리 알려진 휴리스틱 탐색기법 중 하나인 A^* 알고리즘을 활용한 무인차량의 지역경로계획 기법을 제안하고 시뮬레이션을 통해 그 성능을 검증한다. A^* 알고리즘을 이용하여 무인차량에서 실제 활용 가능한 지역경로계획 기법을 구현하기 위해서는 해결해야 할 몇 가지 문제점이 존재한다. 첫째, 지형감지 센서로부터 입력되는 지형 고도값을 활용하여 장애물격자지도를 생성하는 문제 둘째, 센서의 탐지영역에 매핑 되지 않는 GPP 경로점 지향적인 목표점 설정 문제 셋째, 격자지도 기반으로 계획된 경로를 무인차량의 운동성을 고려한 경로로 재생성(Path Smoothing)하는 문제이며 본 논문에서는 각 분야에 대한 해결책을 제시한다. 제안된 지역경로계획 컴포넌트의 성능은 자율주행 성능분석용 M&S(Modeling & Simulation)를 활용하여 각 단위 알고리즘 검증 및 시나리오 기반의 LPP 컴포넌트 통합 검증을 수행한다.

2. 자율 아키텍처와 지역경로계획 구조

가. 무인차량의 자율 아키텍처

무인차량이 주어진 임무의 완수를 목표로 험지 및 야지를 포함하는 거친 환경을 자율적으로 주행하기 위해서는 Fig. 1과 같은 자율 아키텍처를 필요로 한다. 무인차량의 자율 아키텍처는 크게 감지계층(Perception Layer), 판단계층(Cognition Layer) 그리고 동작계층(Action Layer)로 구성되는데, 감지계층을 통해 입력된 센싱 데이터는 판단계층을 구성하는 다양한 지능 관련 컴포넌트에 의해 의사결정에 활용되고 그 결과는 동작계층에 의해 플랫폼의 상태를 제어하게 된다. Fig. 1에서 보이는 컴포넌트들 중 SICK 컴포넌트, LRF World Modeller 컴포넌트 그리고 LRF Map Builder 컴포넌트는 감지계층에 해당되며, DVGM(Directional

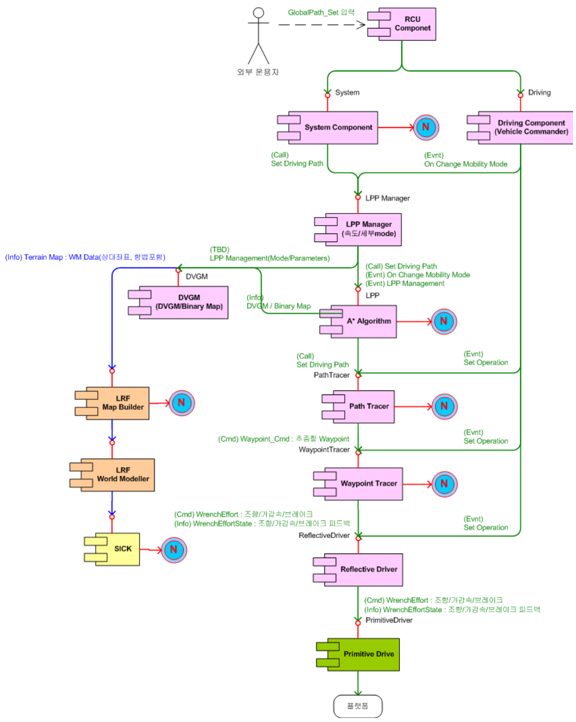

Fig. 1. 무인차량의 자율아키텍처

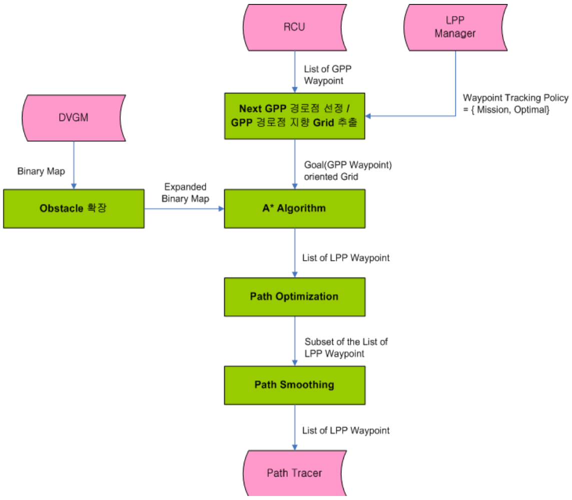

Fig. 2. LPP 컴포넌트 구성 모듈

Velocity Grid Map) 컴포넌트, LPP Manager 컴포넌트 그리고 LPP(A^* Algorithm) 컴포넌트는 판단계층에 해당되며, Path Tracer 컴포넌트, Waypoint Tracer 컴포넌트, Reflective Driver 컴포넌트 그리고 Primitive Driver 컴포넌트는 동작계층에 포함된다.

나. 지역경로계획 컴포넌트 구조

본 논문에서는 판단계층을 구성하는 컴포넌트들 중 A^* 알고리즘 기반의 LPP 컴포넌트에 대해 논의한다. A^* 알고리즘은 가장 널리 알려진 휴리스틱 탐색 기법으로 '언제나 최적의 경로를 산출한다'는 허용성(Admissibility)이 검증된 알고리즘이다[8]. LPP 컴포넌트는 Fig. 2에서처럼 5개의 대표적인 모듈로 구성되는데, 각각의 모듈은 A^* 알고리즘을 활용한 지역경로계획을 위해 구현되어야 할 각 기능을 의미한다.

Next GPP 경로점 선정 및 GPP 경로점 지향 격자 추출 모듈에서는 GPP로부터 입력받은 경로점 집합에서 LPP가 현재 추종해야할 다음 경로점을 선정한 후 이를 지향하는 특정 $G_{(x,\ y)}$를 장애물격자지도에서 추출한다. 장애물 확장 모듈은 센서로부터 입력되는 지형 고도값을 활용하여 DVGM 컴포넌트에서 생성한 장애물[9]의 크기를 실제보다 특정거리만큼 확장하는 역할을 수행한다. 장애물 크기를 확장하여 가상장애물[10]을 생성하는 이유는 무인차량의 운동성으로 인해 회전을 위한 최소 차량길이 반만큼의 여유 공간이 요구되기 때문이다. A^* Algorithm 모듈은 언급한 두 모듈의 결과를 입력받아 목표점까지의 최적 경로를 산출하게 되는데, A^* 알고리즘은 기본적으로 많이 알려져 있기 때문에 본 논문에서는 알고리즘 자체를 언급하지 않는다. Path Optimization 모듈 및 Path Smoothing 모듈은 격자 기반의 A^* Algorithm 모듈이 생성한 추종성이 떨어지는 형태의 경로를 무인차량의 운동성을 고려한 경로로 재생성하여 동작계층인 PathTracer 컴포넌트에 전달한다.

3. 무인차량의 지역경로계획 컴포넌트

본 장에서는 무인차량의 지역경로계획 컴포넌트를 구성하는 각각의 모듈에서 사용되는 세 가지 알고리즘에 대해 논한다. 첫 번째는 'GPP 경로점 지향 격자 추출' 모듈의 기능을 포함하는 이동경비 관점의 최소 지역경로 산출 알고리즘, 두 번째는 장애물 격자지도 기반으로 산출된 A^* 알고리즘의 결과를 보다 최적화 하는 알고리즘, 마지막으로 최적화된 경로를 무인차량의 운동성을 고려한 추종 가능한 형태로 재생성하는 알고리즘을 소개한다.

가. 표기법(Notation) 및 정의(Definition)

지역경로계획 컴포넌트에서 사용되는 세 알고리즘에 대한 정확하고 일관성 있는 논의를 위해 알고리즘 기술에 사용되는 표기법을 먼저 정의할 필요가 있는데 Table 1은 이를 보여준다. 알고리즘 기술에 사용되는 장애물 격자지도는 로컬 좌표계를 기준으로 한다.

나. 최소경비 지역경로 산출 알고리즘

최소경비 지역경로 산출 알고리즘은 Fig. 3의 의사코드(Pseudo-Code)에 보이듯 GPP 경로점을 지향하는 장애물 격자지도상의 특정격자인 *Goal*을 선정하는 함수와 이동거리 경비관점의 최소경로를 산출하는 함수로 구성된다. GPP에서 사용하는 격자지도는 통상적으로 50～200m정도의 크기로 구성되기 때문에 GPP에서 생성한 경로점도 동일한 간격을 가지게 된다. 이러한 이유로 대략 50m이내의 크기로 생성되는 LPP의 장애물 격자지도상에 항상 GPP 경로점이 위치하지 못하게 되며, 이는 *Goal*을 반드시 필요로 하는 A^* 알고리즘을 실행시킬 수 없는 문제를 발생시킨다. 따라서 A^* 알고리즘 기반의 LPP를 위해서는 GPP의 경로점을 지향하는 장애물 격자지도상의 *Goal*을 반드시 선정할 필요가 있다. 또한 LPP의 경로산출 최적성은 바로 이 *Goal* 선

Table 1. 알고리즘 기술용 표기법 및 정의

Notation	Definition
$G(x, y)$	Binary Map의 (x, y)에 놓인 격자
(x_0, y_0)	BM에서 무인차량의 현재 위치로 G(0, 0)
(x_{GPP}, y_{GPP})	BM에서 GPP 경로점의 위치
ΔBM	BM의 무인차량 진행방향 크기
ΔG	BM의 한 격자 크기
CP_x, CP_y	GPP 경로점 지향적인 BM상의 격자 위치
$F(n)$	A^* 알고리즘의 평가함수로 $F(n)=g(n)+h(n)$
w_{factor}	Penalty Weighting Factor
w_{num}	Penalty를 적용 할 양쪽 격자의 개수 차이
P_{num}	생성된 Path를 구성하는 포인트(격자)의 개수
$P_i(x, y)$	Path를 구성하는 i번째 포인트 G(x, y)
$P'_i(x, y)$	현재까지 smoothing된 path를 구성하는 포인트들 중 i번째 포인트 G(x, y)
x_{PS}, y_{PS}	$P_S(x, y)$의 x & y 좌표값
x_{PE}, y_{PE}	$P_E(x, y)$의 x & y 좌표값

정 능력에 상당부분 의존성이 있는데 Fig. 4에 그 이유가 잘 보이고 있다.

Fig. 4a의 경우 장애물 격자지도밖에 존재하는 GPP 경로점인 $G(x_{GPP}, y_{GPP})$에 지향적인 G(x, y)는 검은색으로 표시된 격자이지만, 해당격자에 장애물이 존재하므로 *Goal*이 될 수가 없다. 따라서 그림에 표시된 ①번 또는 ②번 격자가 *Goal*이 될 수 있는 후보격자인데, 제안한 알고리즘에서는 $G(x_0, y_0)$가 존재하는 면을 제외한 장애물 격자지도의 세 모서리를 구성하는 모든 격자의 F(n)값을 조사하여 최소값을 가지는 G(x, y)를 *Goal*로 선정하게 되어 ①번 격자가 *Goal*이 된다. 이렇게 A^* 알고리즘의 F(n) 함수를 이용하는 이유는 *Goal* 선정에 실패한 검은색 격자(*FailedGoal*)와의 최소거리를 기준으로 *Goal*을 선정할 경우 ②번 격자가 *Goal*로 선정되게 되는데 이는 장애물을 우회하는 높은 이동경비를 가지는 경로생성을 유발하여 LPP의 최적성이 감소되기 때문이다. 또한 Fig. 4b의 경우 $G(x_{GPP}, y_{GPP})$에 지향적인 검은색 격자 G(x, y)는 도달할 수 없는 격자로 역시 *FailedGoal*이 된다. 제안된 알고리즘에 의해 세 모서리를 구성하는 모든 격자들 중 A^* 알고리즘의 *CLOSED*에 포함되는 격자들의 F(n)값을 조사하여 최소값을 가지는 격자를 *Goal*로 선정한다. *CLOSED*를 활용하는 이유는 도달할 수 없는 지역인 'Section B'에 포함되는 격자가 *Goal*로 선정되는 것을 막기 위함이다.

```
HeadingAngle(G(x0, y0), G(xGPP, yGPP))
    return tan^-1(xGPP / yGPP);

FindGoalPosition()
    IF(HGPP < -90) set G(-ΔBM, 0) to Goal;
    ELSE IF(HGPP > 90) set G(ΔBM, 0) to Goal;
    ELSE IF(HGPP >= -45 AND HGPP <= 45)
        CPx = round(tan(HGPP) * ΔBM);
        set G(CPx, ΔBM) to Goal;
    ELSE IF((HGPP >= -90 AND HGPP < -45) OR
     (HGPP > 45 AND HGPP <= 90))
        CPy = round(ΔBM / tan(HGPP));
        set G(ΔBM, CPy) to Goal;

PenaltyWeighting()
    WHILE each G(x, y) in opposite side of
     pre_heading on the basis of FailedGoal
        F(n) = F(n) + (Wfactor * Wnum);

ComputeShortestPath()
    HGPP = HeadingAngle(G(x0, y0), G(xGPP, yGPP));
    FindGoalPosition();
    AstarAlgorithm(G(x0, y0), Goal);
    IF fail to compute shortest path
        PenaltyWeighting();
        find G(x, y) having smallest F(n) from
         CLOSED in edges of BinaryMap;
        set G(x, y) to Goal;
        AstarAlgorithm(G(x0, y0), Goal);
```

Fig. 3. 최소경비 지역경로 산출 알고리즘 의사코드

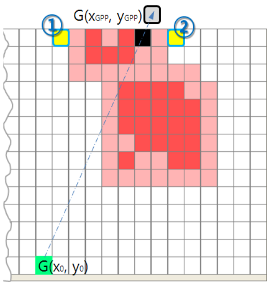

Fig. 4a. *FailedGoal* Ⅰ

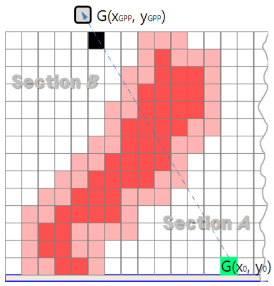

Fig. 4b. *FailedGoal* Ⅱ

LPP의 최적성에 영향을 미치는 *Goal* 선정의 중요성에 대한 또 다른 예를 Fig. 4a에서 확인할 수 있다. 이전의 계산시간에서 ②번 격자처럼 오른쪽 편에 존재하는 *Goal*을 기준으로 LPP를 수행하였다고 가정한 후, Fig. 4a에서 LPP를 수행하면 알고리즘에 의해 ①번 격자를 *Goal*로 선정하게 되어 무인차량이 지그재그형태로 주행하는 문제가 발생하게 된다. 이러한 문제를 해결하기 위해 함수 PenaltyWeighting()에서는 세 모서리를 구성하는 모든 격자들 중 *FailedGoal*을 기준으로 이전 진행방향의 반대편 격자들에 페널티를 부여한다. 알고리즘 11번 라인의 w_{num}값을 '3'으로 설정하면 Fig. 4a에서 ①번이 아닌 ②번 격자를 *Goal*로 선정하여 지역경로계획을 수행할 수 있게 된다.

다. Path Optimization 알고리즘

A^* 알고리즘을 이용하여 LPP를 수행하는 최소경비 지역경로 산출 알고리즘은 격자지도를 활용한다는 특수성으로 인해 무인차량이 선택할 수 있는 조향 값이 45도 단위로 한정될 수밖에 없는 문제점을 지닌다. 이러한 이유로 주어진 *Goal*까지 생성된 지역경로는 실질적인 최단거리의 경로가 되지 못하며, 또한 잦은 조향으로 인해 추종성의 저하를 초래하게 된다. 제안하는 Path Optimization 알고리즘은 이러한 문제를 해결하기 위한 알고리즘으로, 계획된 지역경로를 구성하는 포인트들의 집합에서 각각의 포인트 $P_i(x, y)$에 대해 다른 포인트들과 생성 가능한 모든 경우의 직선을 추출한 후 장애물과 접촉하지 않으면서 길이가 가장 긴 직선을 선정하여 두 포인트 사이의 경로를 격자단위를 무시한 직선화를 통해 경로 최적화를 수행한다. Fig. 5는 해당 알고리즘의 의사코드이다.

라. Path Smoothing 알고리즘

Path Optimization 알고리즘을 통해 지역경로를 최적화하여도 장애물의 배치에 따라서는 최대 90도의 조향이 요구되는 경로의 포인트가 존재할 수 있다. 다시 말해 최적화된 경로를 무인차량의 운동성을 고려한 추종 가능한 형태의 경로로 재생성하는 알고리즘이 요구된다는 의미이다. 제안하는 Path Smoothing 알고리즘은 Bezier Spline 알고리즘을 활용하는데 이는 3차 Spline 알고리즘 중 가장 보편적이고 곡선 제어가 용이한 알고리즘으로, 네 개의 제어점을 사용하여 첫 번째 포인트에서 네 번째 포인트를 잇는 곡선을 산출한다.

Path Optimization 알고리즘을 적용하여 직선화된 경로를 구성하는 모든 포인트 $P_i(x, y)$들은 무인차량의 조향이 요구되는 *EDGE*가 되는데 이는 Bezier Spline 알고리즘을 적용하는 기준점이 된다. *EDGE*를 기준으로 현재까지 Smoothing된 경로에서 마지막 두 포인트

```
BlockingCheck(P_S(x, y), P_E(x, y))
01. slope = (y_PE - y_PS) / (x_PE - x_PS);
02. FOR(i = x_PS to x_PE)
03.     y_lower = round(y_PS + (i - x_PS - 0.5)*slope);
04.     y_upper = round(y_PS + (i - x_PS + 0.5)*slope);
05.     IF(i == x_PS) y_lower = y_PS;
06.     IF(i == x_PE) y_upper = y_PE;
07.     FOR(j = y_lower to y_upper)
08.         IF(G(i, j) == obstacle)
09.             return BLOCKED; break;
10. return NOT_BLOCKED;

PathOptimization()
11. FOR(i = 0 to i < P_num)
12.     FOR(j = P_num to j > i)
13.         IF BlockingCheck(P_i(x, y), P_j(x ,y)) ==
             NOT_BLOCKED
14.             remove points between P_i and P_j;
15.             i = j; break;
```

Fig. 5. Path Optimization 알고리즘 의사코드

```
Initialize()
01. Spline(P_0(x, y), G(0, 1), P_2(x, y), P_3(x, y));
02. tag P_3(x, y) with EDGE;

Spline(P_0, P_1, P_2, P_3);
03. S_num = sqrt((x_P3 - x_P0)^2 + (y_P3 - y_P0)^2) / S_size;
04. FOR(n = 0 to S_num)
05.     x_SPLINE = x_P0*(1 - n)^3 + 3*x_P1*n(1 - n)^2 +
                3*x_P2*n^2*(1 - n) + x_P3*n^3;
06.     y_SPLINE = y_P0*(1 - n)^3 + 3*y_P1*n(1 - n)^2 +
                3*y_P2*n^2*(1 - n) + y_P3*n^3;
07.     DrawPoint(x_SPLINE, y_SPLINE);

PathSmoothing()
08. tag all points ∈ optimized path with EDGE;
09. segment the optimized path into ΔG;
10. Initialize();
11. FOR(n = 3 to P_num)
12.     IF P_n(x, y) == EDGE
13.         Spline(P'_n-2(x, y), P'_n-1(x, y), P_n+1(x, y),
                P_n+2(x, y));
14.         n = n + 2;
```

Fig. 6. Path Smoothing 알고리즘 의사코드

를 추출하고 세분화(Segmentation)시킨 최적화 경로에서 다음 두 포인트를 추출하여 Spline 알고리즘에 입력한 후 첫 번째 포인트와 네 번째 포인트를 반드시 지나는 추종성이 확보된 경로를 재 생성한다. Fig. 6은 해당 알고리즘의 의사코드이다.

4. 시뮬레이션 결과 및 분석

가. LPP 성능분석을 위한 자율 M&S

본 논문에서 제안된 장애물 격자지도 기반의 지역경로계획 컴포넌트의 성능검증 및 분석을 위해 국방과학연구소에서 개발한 자율주행 성능분석용 M&S (Modeling & Simulation)를 활용하였다. 자율주행 성능분석을 위한 가상환경은 개방형 아키텍처를 활용한 분산 네트워크 기반의 가상 모델링 환경 및 실시간 시뮬레이션 환경으로 구축되었으며 Fig. 7과 같이 6개의 하위 구성 그룹인 가상환경 생성 그룹, 자율처리 그룹, 가상센서 처리 그룹, 가시화 그룹, 분석 및 로깅 그룹 그리고 HILS(Hardware In the Loop System) 그룹을 가진다[11]. 무인차량이 가상의 임무를 수행할 수 있도록 사용자가 운용시나리오 및 임무를 생성하면 LPP 컴포넌트가 포함된 자율 처리 그룹에서는 주어진 시나리오에 따라 주행하면서 가상센서 처리 그룹으로부터 입력되는 통합 월드모델링 정보를 활용하여 지능적인 의사결정을 수행한다. 이렇게 결정된 상태정보는 주행모의 컴포넌트를 통해 시뮬레이션 환경에 피드백 된다.

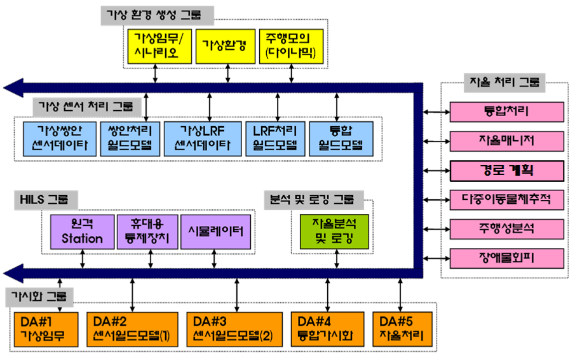

Fig. 7. 자율주행 성능분석용 M&S 구조

나. 최소경비 지역경로 산출 알고리즘 검증

본 절에서는 최소경비 지역경로 산출 알고리즘 중 GPP 경로점을 지향하는 장애물 격자지도상의 특정격자인 *Goal*을 선정하는 함수의 성능을 검증한다. Fig. 8a를 보면 GPP 경로점을 지향하는 회색의 격자에 장애물이 존재하여 해당 격자는 *FailedGoal*이 되며, 제안한 알고리즘에 의해 이를 대신할 수 있는 후보격자는 ①번 격자 및 ②번 격자로 축약된다. Table 2에서 두 후보격자의 평가함수값 F(n)을 비교해보면 ②번 격자가 '25.1'로 더욱 적어 해당격자로 이동함에 있어 보다 최소경비가 소모됨을 의미하므로 해당 격자가 새로운 *Goal*로 선정되어 지역경로계획을 수행한다. 또한 Fig. 8b를 보면 GPP 경로점을 지향하는 회색의 격자에 장애물이 존재하지는 않지만 도달 불가능한 격자이므로 *FailedGoal*이 되며, 제안한 알고리즘에 의해 세 모서리를 구성하는 모든 격자들 중 *CLOSED*에 포함되는 격자들의 F(n)값을 조사하여 최소값을 가지는 격자를 새로운 *Goal*로 선정하여 지역경로계획을 수행함을 확인할 수 있다.

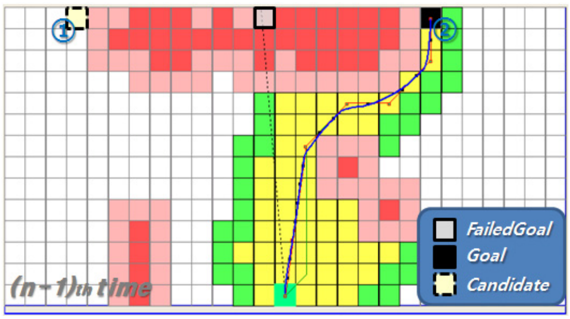

Fig. 8a. 함수 FindGoalPosition()의 검증 Case Ⅰ

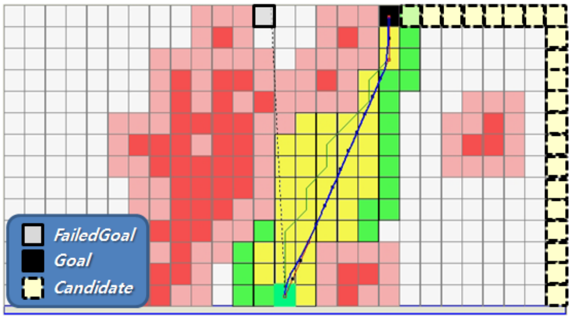

Fig. 8b. 함수 FindGoalPosition()의 검증 Case Ⅱ

GPP에서 생성한 경로점을 지향하는 *Goal* 선정에 있어 함수 PenaltyWeighting()의 역할이 Fig. 9에서 검증된다. $(n\text{-}1)_{th}$ 계산시간에 Fig. 8a처럼 *FailedGoal*을 기준

으로 오른쪽 편에 존재하는 *Goal*에 대해 LPP를 수행하였다고 가정한 후, $(n)_{th}$ 계산시간에 Fig. 9의 장애물 격자지도에서 함수 PenaltyWeighting() 없이 *Goal*을 선정한다면 Table 2의 F(n)값 비교에 의해 Fig. 9a처럼 더욱 작은 값을 가지는 ①번 격자를 *Goal*로 선정하게 되어 무인차량이 지그재그형태로 주행하는 문제가 발생하게 된다. 반면 함수 PenaltyWeighting()을 적용하게 되면 *FailedGoal*을 기준으로 왼쪽에 존재하는 후보격자 ①번의 F(n)값이 페널티로 인해 '28.5'로 상승하여 Fig. 9b처럼 ②번 격자가 *Goal*로 선정되어 이러한 문제가 해결됨을 확인할 수 있다.

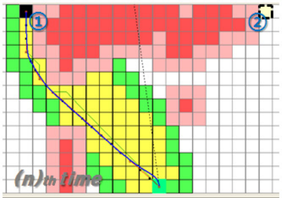

Fig. 9a. Penalty 미적용

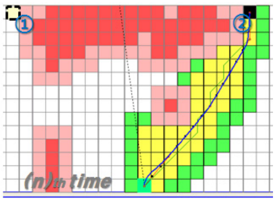

Fig. 9b. Penalty 적용

Table 2. Penalty 적용/미적용 시 F(n)값

	Fig. 8a		Fig. 9a		Fig. 9b	
	격자①	격자②	격자①	격자②	격자①	격자②
F(n)	26.1	25.1	25.7	26.3	25.7	26.3
g(n)	17.1	17.1	17.7	16.3	17.7	16.3
h(n)	9.0	8.0	8.0	10.0	8.0	10.0
penalty 적용	-	-	-	-	28.5	26.3

다. Path Optimization & Smoothing 검증

본 절에서는 격자지도를 활용함으로서 생기는 문제점을 해결하기 위한 Path Optimization 알고리즘과 A^* 알고리즘에 의해 생성된 지역경로를 무인차량의 운동성을 고려한 추종 가능한 형태의 경로로 재생성하는 Path Smoothing 알고리즘에 대해 검증한다. Fig. 10에서 Path Optimization 알고리즘의 수행 결과를 확인할 수 있는데, $G(x_0, y_0)$와 *Goal* 사이의 가는 직선은 최소경비 지역경로 산출 알고리즘인 A^*로 생성된 최초의 경로이며 굵은 점선은 Path Optimization 알고리즘을 적용하여 최적화시킨 지역경로를 의미한다. 조향에 대한 경비를 포함시키지 않고 한 격자를 수평/수직방향으로 이동하는데 소비되는 경비를 '1' eu(energy unit)라고 정의하면 A^* 알고리즘에 의해 생성된 최초의 경로는 '18.6' eu가 소비되는 반면, 해당 알고리즘을 적용하여 최적화 시킨 경로는 '17.6' eu가 소비되어 이동거리 경비 관점에서 보다 최적화되었음을 확인할 수 있다.

Fig. 11에서는 Path Smoothing 알고리즘의 수행 결과를 확인할 수 있는데 점선의 원으로 표시된 부분은 무인차량의 조향이 요구되는 *EDGE*로서 이는 경로 재생성을 위한 Bezier Spline 알고리즘을 적용하는 기준점이 된다. 그림의 굵은 실선은 Path Smoothing을 통해 재 생성된 지역경로로, 차량의 주행특성을 고려하지 않은 격자기반의 조향으로 구성된 최초의 경로에서 차량의 운동성이 고려된 추종 가능한 조향으로 구성된 경로로 변경되었음을 확인할 수 있다.

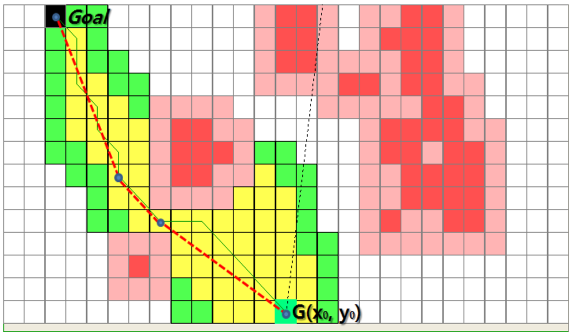

Fig. 10. Path Optimization 알고리즘의 결과

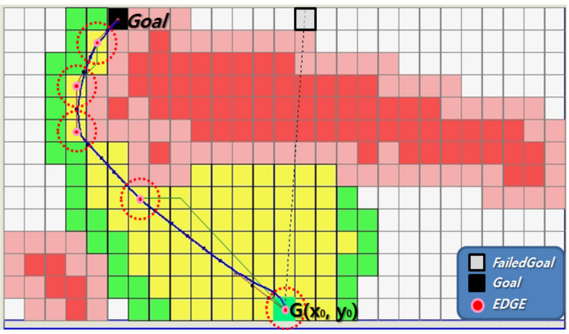

Fig. 11. Path Smoothing 알고리즘의 결과

라. 시나리오 기반의 LPP 컴포넌트 통합 검증

본 절에서는 앞 장에서 언급한 각 알고리즘들을 통합적으로 검증할 수 있는 시나리오를 생성하여 시나리오 기반의 LPP 컴포넌트 통합 검증을 수행한다. 시

나리오는 특정지역의 DEM(Digital Elevation Map)상에서 정의되었으며 Table 3에 보이듯 세 개의 개별 시나리오로 구성되는데 각 시나리오는 수십 m의 간격을 가지는 GPP 경로점들을 가진다. s1은 드럼통 및 벽과 같은 고정 장애물로 구성된 시나리오이며, s2는 벽으로 생성된 좁은 통로로 이루어진 시나리오이며 마지막으로 s3은 드럼통, 벽, 나무와 같은 고정 장애물과 타원의 주행경로를 가지는 자동차와 같은 이동 장애물로 구성된 시나리오이다. Fig. 12는 시나리오 s1에 대한 주행결과를 보여주는데 경로점과 경로점 사이의 점선은 GPP에서 생성한 전역경로를 의미하며, 실선은 무인차량에 탑재된 LPP 컴포넌트에 의해 주행된 실제 이동경로를 의미한다. Fig. 13은 시나리오 s2에 대한 자율주행 결과로 벽으로 구성된 좁은 통로에서 지역 최소점 문제에 빠지지 않고 안전하게 통로를 빠져나와 목표점까지 주행함을 확인할 수 있다. Fig. 14는 시나리오 s3에 대한 자율주행 결과로 다양한 고정 장애물과 이동 장애물로 구성된 시나리오 상에서 장애물과의 충돌 없이 지역경로계획을 수행하여 목표점에 도달하였음을 확인할 수 있다.

Table 3. LPP 통합 검증을 위한 시나리오

#	Situation	Description
s1	Static Obst.	Drum + Wall Obstacle
s2	Wall Following	Corridor Navigation
s3	Moving Obst.	Static Obst. + Moving Car

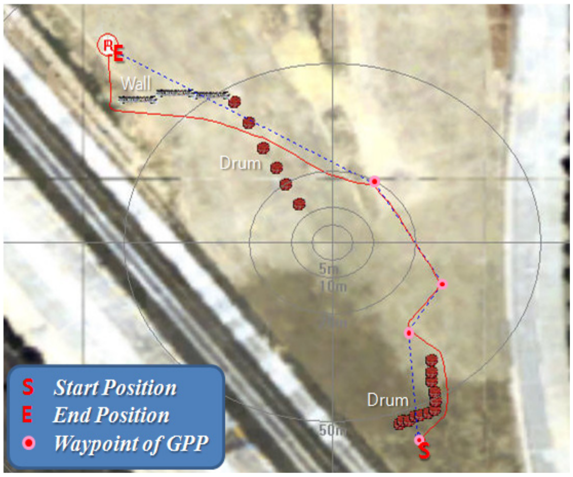

Fig. 12. 시나리오 s1에 대한 LPP 수행 결과

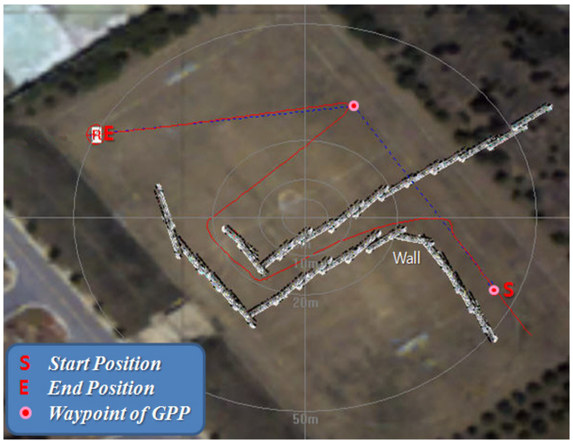

Fig. 13. 시나리오 s2에 대한 LPP 수행 결과

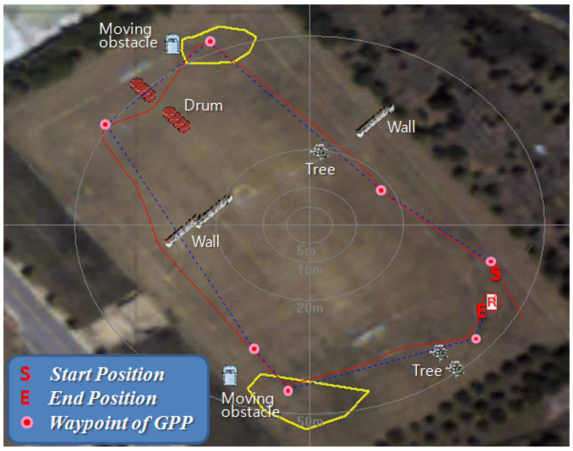

Fig. 14. 시나리오 s3에 대한 LPP 수행 결과

5. 결 론

본 논문에서는 지형감지센서로부터 획득한 월드모델 데이터를 이용하여 생성된 장애물 격자지도(Binary Map) 기반의 무인차량용 지역경로계획 컴포넌트를 제안하였다. 특히 무인차량의 지역경로계획 컴포넌트를 구성하는 각각의 모듈에서 사용되는 세 가지 핵심 알고리즘인 최소경비 지역경로 산출 알고리즘, Path Optimization 알고리즘, 그리고 Path Smoothing 알고리즘을 소개하였다. 최소경비 지역경로 산출 알고리즘에서는 GPP 경로점 지향적인 *Goal*을 선정하는 효율적인 방법을 제안하였으며 또한, LPP의 경로산출 최적성을 저하시키는 지그재그 형태의 *Goal* 선정 문제 해

결을 위해 PenaltyWeighting() 함수를 적용하였다. Path Optimization 알고리즘에서는 45도 단위의 한정된 조향 문제와 이로 인해 발생하는 잦은 조향 문제의 해결을 통해 산출된 경로를 이동경비 관점에서 보다 최적화 시켰으며, Path Smoothing 알고리즘은 Optimization 알고리즘을 통해 최적화된 경로에 무인차량의 운동성을 고려하여 보다 추종성이 향상된 지역경로를 재생성할 수 있도록 하였다.

제안된 지역경로계획 컴포넌트의 성능은 자율주행 성능분석용 M&S를 활용하여 각 단위 알고리즘 검증 및 시나리오 기반의 LPP 컴포넌트 통합 검증을 통해 확인하였으며, 이동거리 경비 관점에서 목표점까지의 효율적이고 안전한 지역경로계획이 수행됨을 시뮬레이션 결과를 통해 확인하였다.

Reference

[1] J. Giesbrecht, J. Collier, G. Broten, S. Monckton, and D. Mackay, “A Navigatino and Decision Making Architecture for Unmanned Ground Vehicles”, DRDC Suffield TM 2007-300, 2007.

[2] C. Howie, L. Kevin, H. Seth, K. George, B. Wolfram, K. Lydia, and T. Sebastian, Principles of Robot Motion : Theory, Algorithms, and Implementations, 2005.

[3] Borenstein, J., and Koren, Y., “Real-time Obstacle Avoidance for Fast Mobile Robots”, IEEE Transactions on System, Man, and Cybernetics, Vol. 19, Oct., 1989.

[4] Borenstein, J., and Koren, Y., “The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots”, IEEE Journal of Robotics and Automation, Vol. 7, June, 1991.

[5] Borenstein, J., and Koren, Y., “Real-Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments”, IEEE Internatinoal Conference of Robotics and Automation, pp. 572～577, 1990.

[6] Lee, Y. I. and Kim, Y. G., “An Intelligent Collision Avoidance System for AUVs using Fuzzy Relational Products”, Information Sciences, 2004.

[7] Stentz, A., “Optimal and Efficient Path Planning for Partially-Known Environments”, Proc. of IEEE International Conference on Robotics and Automation, pp. 3310～3317, 1994.

[8] Nilsson, N. J., Principles of Artificial Intelligence, Tioga Publishing Co., 1980.

[9] 이영일, 이호주, 지태영, “무인차량의 주행성분석을 위한 방향별 속도지도 생성”, 한국군사과학기술학회지, Vol. 12, No. 5, pp. 549～556, Oct., 2009.

[10] Lee, Y. I. and Kim, Y. G., “A Collision Avoidance System for Autonomous Ship Using Fuzzy Relational Products and COLREGs”, Lecture Notes in Computer Science, Intelligent Data Engineering and Automated Learning-IDEAL2004, pp. 247～252, 2004.

[11] 안명길, 이석재, 박용운, 고정호, “자율주행 성능분석을 위한 가상환경 및 센서 모델링 기법 연구”, 전자공학회 논문지, Vol. 45, pp. 10～15, 2008.